Molecular dynamics simulation methods revised

Bekker, Hendrik

2 AN EFFICIENT NON-BONDED FORCE ALGORITHM

A notation is introduced and used to transform a conventional specification of the non-bonded force and virial algorithm in the case of periodic boundary conditions into an alternative specification. The implementation of the transformed specification is simpler and typically a factor of 1.5 faster than a conventional implementation. Moreover, it is generic with respect to the shape of the simulated system, i.e. the same routines can be used to handle triclinic boxes, truncated octahedron boxes etc. An implementation of this method is presented, and the speed achieved on various machines is given. The essence of the new method is that the number of calculations of image particle positions is strongly reduced.

2.1 Introduction

Conceptually, the M.D. simulationtechnique is simple: the time development of a many particle system is numerically evaluated by integrating the equations of motion. However, the inevitable introduction of time saving concepts as a cut-off radius, periodic boundary conditions, the nearest image criterion, and non-trivial box shapes makes realistic implementations complex, and makes it difficult to think in a clear way about alternative algorithms.

In this chapter we will try to master this complexity by starting at the specification level and by introducing a notation which makes it easy to derive alternative specifications in a formal way from an initial specification. This proves to be simple and clear, and results in a surprising new non-bonded force (NBF) algorithm and a new virial algorithm. The new algorithms are generic with respect to the shape

of the computational box, that is, the same NBF and virial algorithm may be used for the triclinic box, the truncated octahedron, and three more box shapes which can be stacked in a space filling way. Moreover, the implementation of the new algorithms, tested on a wide range of machines, proves to be faster than conventional implementations by at least a factor of 1.5.

Besides these tangible results, our derivations also have the potential of giving a solid base to existing and new algorithms of other parts of the M.D. algorithms such as neighbour searching (NS) and bonded force calculations. In this chapter we will confine ourselves to the non-bonded and virial calculations, and do not address neighbour searching.

As will be clear from the foregoing, we do not arrive at the new algorithms by transformations on the implementation level but by *transforming a specification* which is subsequently implemented rather straightforwardly in software. Three conditions should be fulfilled to make this method feasible: a suitable formal notation should be introduced, a correct initial specification should be available, and a set of transformation rules should be given. In [1] we used a notation which is rather clumsy, and which did not enable us to derive the results we present in this chapter. In contrast, the notation we use in this chapter lends itself very well for creating and manipulating specifications, and yet is simple. Also, in [1] we did not manipulate a specification but gave a proof at the operational level. In contrast, we now start with the specifications. The derivation of a correct specification for the NBF calculations is rather straightforward. Although a bit more complex this also is the case for the virial calculation. The set of transformation rules we apply is simple and their correctness is easily verified with a two-particle M.D. system.

Following the usual M.D. practice, involving the minimum image convention, we assume that the cut-off distance R_{co} is such that particles in the central box only interact with particles which are in the central box and in directly adjacent boxes. The minimum image convention requires that every particle interacts with every other particle at most once, implying that R_{co} does not exceed half the length of the smallest image displacement vector. However, the algorithms we derive can be used as well for larger values of R_{co}.

The algorithms derived in this chapter have been implemented on the GROMACS parallel computer for M.D. simulations [2,3]. However, because the GROMACS implementation also contains other innovations we will not discuss that implementation but a simpler one.

In the next sections of this chapter we will discuss some properties of systems

with Periodic Boundary Conditions (PBC), introduce a notation for particle positions and forces in PBC systems, and derive new force and new virial algorithms. We will discuss an implementation of these algorithms and compare the performance of a conventional implementation with the newly derived implementation. Finally we generalise the theory to other box shapes and discuss some other possible derivations of other parts of the M.D. algorithm.

2.2 Derivations

2.2.1 Notation

To mitigate boundary effects in finite systems, most M.D. simulations are done on systems with PBC. An M.D. system with PBC consists of a central computational box with N particles, surrounded by an infinite number of translated identical image boxes, stacked in a space filling way. In the following we concentrate on the often used generalised cube or *triclinic box*, which is a chunk of space enclosed between six pairwise parallel planes. Other box shapes will be treated in Section 2.3. The box is defined by its three spanning vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$. Every image box is a translated copy of the central box with a translation vector $\mathbf{t}$ given by

$$\mathbf{t} \equiv n_a\,\mathbf{a} + n_b\,\mathbf{b} + n_c\,\mathbf{c} \quad (n_a, n_b, n_c \in \mathbb{Z})\,. \tag{2.1}$$

We define the minimum box size L_{min} as the minimum distance between the parallel planes enclosing the box. Under the minimum-image convention we assume that $R_{co} < \frac{1}{2}L_{min}$. To calculate the behaviour of the infinite system, only the behaviour of particles in the central box has to be calculated. When $R_{co} < \frac{1}{2}L_{min}$, particles in the central box can only be influenced by particles in the central box and particles in the 26 directly adjacent boxes. So by numbering the boxes from -13 to 13 we are able to uniquely identify all boxes and particles we need to refer to. Using n_a, n_b, n_c of (2.1) we define the boxnumber k as

$$k = 9n_a + 3n_b + n_c \quad -1 \le n_a, n_b, n_c \le 1\,, \tag{2.2}$$

so the central box has boxnumber 0. In Figure 2.1 a 2-D PBC system is shown with $-4 \le k \le 4$. Note that the box opposite to box k with respect to the central box has boxnumber $-k$.

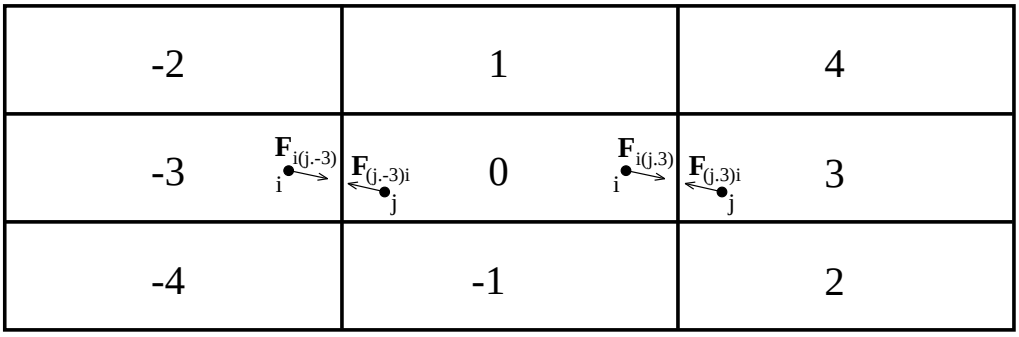

FIGURE 2.1 2-D PBC system with boxes numbered -4 to 4, and with the relations between the four forces as given by equation (2.4). Note that box k is opposite of box $-k$.

We will use the notation $(j.k)$ to indicate the particle with number j in box k. A single particle number j will indicate the particle with number j in box 0, that is, particle j is another notation for particle $(j.0)$.

The position of particle $(j.k)$ is denoted by $\mathbf{r}_{(j.k)}$ and is given by the position of its parent particle in the central box plus the translation vector $\mathbf{t}_k$ of the box k, so

$$\mathbf{r}_{(j.k)} = \mathbf{r}_j + \mathbf{t}_k \; . \tag{2.3}$$

The force exerted on particle $(j.k)$ by particle i in the central box is denoted by $\mathbf{F}_{(j.k)i}$. In the following we will often use the equalities (see also Figure 2.1)

$$\mathbf{F}_{i(j.k)} = \mathbf{F}_{(i.-k)j} = -\mathbf{F}_{(j.k)i} = -\mathbf{F}_{j(i.-k)} \; , \tag{2.4}$$

and

$$\mathbf{t}_k = -\mathbf{t}_{-k} \; . \tag{2.5}$$

2.2.2 Force derivations

In the usual implementation of a system of N particles the total force $\mathbf{F}_i$ on particle i is calculated as

$$\mathbf{F}_i = \sum_{j=1}^{N} \sum_{k=-13}^{13} \mathbf{F}_{i(j.k)} \; . \tag{2.6}$$

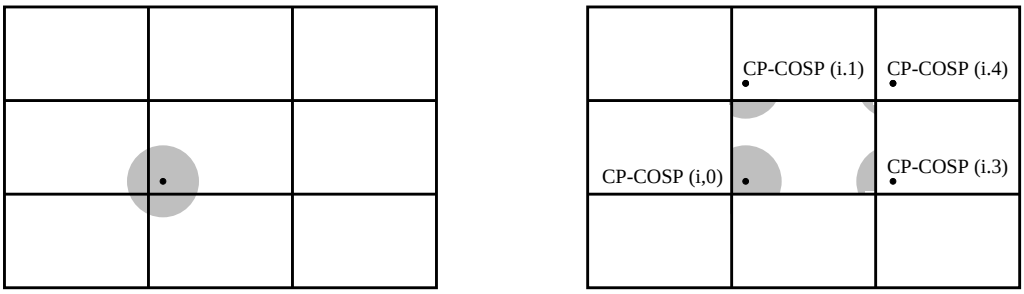

FIGURE 2.2 (a) Cut-off sphere in conventional way; dot is CP. (b) Cut-off sphere partitioned according to (2.8); dots are CP-COSPs.

Although we include the full sum over k, under the minimum image convention for each j only one k will contribute to the sum; all other image forces vanish because of the cut-off condition.

Using (2.4), equation (2.6) can be rewritten giving

$$\mathbf{F}_i = \sum_{j=1}^{N} \sum_{k=-13}^{13} \mathbf{F}_{(i.-k)j} \,. \tag{2.7}$$

When k runs from -13 to 13, $-k$ runs from 13 to -13. Because addition is commutative the order in which k runs through the interval $[13,-13]$ does not matter, so for $\mathbf{F}_i$ we may write

$$\mathbf{F}_i = \sum_{j=1}^{N} \sum_{k=-13}^{13} \mathbf{F}_{(i.k)j} \,. \tag{2.8}$$

Let us elaborate on this expression because it is a central idea of this chapter. What (2.8) means is that the total force on particle i is calculated as a sum of forces on images of i (including the 'null image' in the central box) exerted only by particles in the central box. It is as if the usual cut-off sphere is partitioned by (image) box boundaries, and every cut-off sphere partition (COSP) is translated into the central box. With those COSPs which actually are translated, the central particle of the original cut-off sphere is also translated, resulting in central particles of partial cut-off spheres outside the central box. In Figure 2.2 both ways of calculating $\mathbf{F}_i$ are shown.

For further discussions we introduce the following notions. We call the conventional unpartitioned cut-off sphere 'Cut-Off Sphere' (COS), and partition the conventional cut-off sphere into 'cut-off sphere partitions' (COSPs). Also we keep calling the central particle $(i.0)$ of the conventional cut-off sphere 'central particle' (CP), and call the central particle of a COSP, 'central particle of cut-off sphere partition' (CP-COSP). With neighbours of a CP we will mean all particles, both in the central box and image boxes, within R_{co} of that CP. With neighbours of a CP-COSP we mean all particles *in the central box* within R_{co} of that CP-COSP.

Because $R_{co} < \frac{1}{2}L_{min}$ the neighbours of a given CP are located in at most eight boxes (including the central box). Therefore, every particle in the central box is split in at most eight CP-COSPs, each with its own neighbourlist. The total number of neighbours in these neighbourlists is equal to the number of neighbours in the neighbourlist of the corresponding CP, so the total number of NBF interaction calculations is not influenced by the use of the COSP method.

2.2.3 Virial derivations

We define the virial tensor product $\mathbf{W}_{ij}$ of an interacting pair of particles as

$$\mathbf{W}_{ij} \equiv \mathbf{r}_{ij} \otimes \mathbf{F}_{ij} \,, \tag{2.9}$$

where the tensorial direct product $\otimes$ is defined as

$$(\mathbf{u} \otimes \mathbf{v})_{\alpha\beta} = (\mathbf{u})_{\alpha}(\mathbf{v})_{\beta} \,, \quad \alpha, \beta = x, y, z \tag{2.10}$$

and $\mathbf{r}_{ij} \equiv \mathbf{r}_i - \mathbf{r}_j$.

As Erpenbeck and Wood [4] have shown (using a different notation),

$$-\frac{1}{2}\mathbf{W} \equiv -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{i-1}\sum_{k=-13}^{13} \mathbf{r}_{i(j.k)} \otimes \mathbf{F}_{i(j.k)} \tag{2.11}$$

represents the internal Clausius virial in a periodic system. This expression takes each interaction for which i and j are both in the central box, into account only once. Interactions over the boundary of the central box occur pairwise (see Figure 2.1). Of these interactions only one is taken into account by (2.11).

Equation (2.11) can be used to calculate the instantaneous pressure tensor $\mathbf{P}$ of the M.D. system as

$$\mathbf{P} = \frac{1}{V}\left(\mathbf{W} + \sum_{i=1}^{N} m_i \mathbf{v}_i \otimes \mathbf{v}_i\right) \,, \tag{2.12}$$

where m_i is the mass and $\mathbf{v}_i$ the velocity of the ith particle, and V the volume of the system.

The straightforward implementation of (2.11) involves its evaluation in the inner loop of the non-bonded force routine, which results in a significant CPU time consumption for this expression. We will therefore investigate how this expensive operation can be transformed in such a way that it can be moved to an outer loop.

Using $\mathbf{r}_{i(j.k)} = \mathbf{r}_i - \mathbf{r}_j - \mathbf{t}_k$, we split $\mathbf{r}_{i(j.k)}$ in (2.11) in three terms, $\mathbf{r}_i$, $\mathbf{r}_j$ and $\mathbf{t}_k$, which gives

$$\begin{aligned}\mathbf{W} &= \sum_{i=1}^{N}\sum_{j=1}^{i-1}\sum_{k=-13}^{13} \mathbf{r}_i \otimes \mathbf{F}_{i(j.k)} + \sum_{i=1}^{N}\sum_{j=1}^{i-1}\sum_{k=-13}^{13} \mathbf{r}_j \otimes \mathbf{F}_{(j.k)i} + \\ &\quad + \sum_{i=1}^{N}\sum_{j=1}^{i-1}\sum_{k=-13}^{13} \mathbf{t}_k \otimes \mathbf{F}_{(j.k)i} \\ &\equiv A + B + C\,. \end{aligned} \tag{2.13}$$

We will now show that the sum of A and B can be written as

$$A + B = \sum_{i=1}^{N} \mathbf{r}_i \otimes \mathbf{F}_i\,. \tag{2.14}$$

This follows from a rewriting of term B. Using $\mathbf{F}_{(j.k)i} = \mathbf{F}_{j(i.-k)}$ (see (2.4)) gives

$$B = \sum_{i=1}^{N}\sum_{j=1}^{i-1}\sum_{k=-13}^{13} \mathbf{r}_j \otimes \mathbf{F}_{j(i.-k)}\,. \tag{2.15}$$

Because k runs through values which are symmetric with respect to 0 we may replace $-k$ by k

$$B = \sum_{i=1}^{N}\sum_{j=1}^{i-1}\sum_{k=-13}^{13} \mathbf{r}_j \otimes \mathbf{F}_{j(i.k)}\,. \tag{2.16}$$

We may of course interchange the names i and j

$$B = \sum_{j=1}^{N}\sum_{i=1}^{j-1}\sum_{k=-13}^{13} \mathbf{r}_i \otimes \mathbf{F}_{i(j.k)}\,. \tag{2.17}$$

Because $(1 \le j \le N) \wedge (1 \le i \le j-1)$ implies $(1 \le i < N) \wedge (i+1 \le j \le N)$, we get $\sum_{j=1}^{N}\sum_{i=1}^{j-1}(i,j) = \sum_{i=1}^{N}\sum_{j=i+1}^{N}(i,j)$. This gives

$$B = \sum_{i=1}^{N}\sum_{j=i+1}^{N}\sum_{k=-13}^{13} \mathbf{r}_i \otimes \mathbf{F}_{i(j.k)}\,. \tag{2.18}$$

Adding A and B and using $\mathbf{F}_{i(i.k)} = 0$ gives

$$A + B = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=-13}^{13} \mathbf{r}_i \otimes \mathbf{F}_{i(j.k)} \,. \tag{2.19}$$

Using $\sum_{j=1}^{N} \sum_{k=-13}^{13} \mathbf{F}_{i(j.k)} = \mathbf{F}_i$ we get the required equation (2.14).

To rewrite C we define $\mathbf{g}_k$ as the total force on box k exerted by all higher numbered particles in the central box

$$\mathbf{g}_k \equiv \sum_{i=1}^{N} \sum_{j=1}^{i-1} \mathbf{F}_{(j.k)i} \,. \tag{2.20}$$

Using this definition C can be written as

$$C = \sum_{k=-13}^{13} \mathbf{t}_k \otimes \mathbf{g}_k \,. \tag{2.21}$$

Adding A, B and C we get

$$\mathbf{W} = \sum_{i=1}^{N} \mathbf{r}_i \otimes \mathbf{F}_i + \sum_{k=-13}^{13} \mathbf{t}_k \otimes \mathbf{g}_k \,. \tag{2.22}$$

The last expression for **W** means that the virial may be calculated as if no periodicity exists (the $\sum_{i=1}^{N} \mathbf{r}_i \otimes \mathbf{F}_i$ part), corrected with the $\sum_{k=-13}^{13} \mathbf{t}_k \otimes \mathbf{g}_k$ part. So, we have now reduced the double sum for **W** to a single sum plus a correction part C, which both can be cheaply evaluated in an outer loop of the NBF routine. Also, the calculation of $\mathbf{g}_k$, required for C, can be done in an outer loop of the NBF routine because, when using the COSP method, for every CP-COSP $(i.k)$ the sum $\sum_{i=1}^{N} \mathbf{F}_{(j.k)i}$ is calculated (2.8). (See the first statement on line 27 of the listing in Section 2.2.3.)

The term C in (2.22) can be written in several ways. Which one is the most practical depends on the way forces are calculated. We will now derive some alternative expressions for C.

By exchanging i and j in (2.20) and rewriting the double sum as was done in going from (2.16) to (2.18) we find that $\mathbf{g}_k$ may also be written as

$$\mathbf{g}_k = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathbf{F}_{(i.k)j} \,. \tag{2.23}$$

The correction term C can also be written as

$$C = \sum_{k=-13}^{13} \mathbf{t}_k \otimes \mathbf{g}_k = \sum_{k=-13}^{13} \mathbf{t}_{-k} \otimes \mathbf{g}_{-k} = \sum_{k=-13}^{13} \mathbf{t}_k \otimes \mathbf{g}'_k \tag{2.24}$$

where $\mathbf{g}'_k \equiv -\mathbf{g}_{-k}$. Using (2.20) and (2.4) gives

$$\mathbf{g}'_k = -\sum_{i=1}^{N}\sum_{j=1}^{i-1}\mathbf{F}_{(j.-k)i} = -\sum_{i=1}^{N}\sum_{j=1}^{i-1}\mathbf{F}_{j(i.k)} = \sum_{i=1}^{N}\sum_{j=1}^{i-1}\mathbf{F}_{(i.k)j}\,. \tag{2.25}$$

A comparison of (2.23) and the last expression in (2.25) shows that the correction term C can also be written as

$$C = \frac{1}{2}\sum_{k=-13}^{13}\mathbf{t}_k \otimes \mathbf{g}''_k\,. \tag{2.26}$$

where

$$\mathbf{g}''_k \equiv \sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{F}_{(i.k)j} = \sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{F}_{(j.k)i}\,. \tag{2.27}$$

This equation truly represents the total force of all particles in the central box on all particles in box k.

2.2.4 Neighbour searching

In this subsection we will discuss the consequences of (2.8) on neighbour searching and the structure of neighbourlists.

When non-bonded forces are calculated in the conventional way, that is with (2.6), for every CP a neighbourlist has to be constructed. When non-bonded forces are calculated in the new way, that is with (2.8), for every CP-COSP a neighbourlist has to be constructed. In contrast to the neighbourlist of a CP, the neighbourlist of a CP-COSP should only contain particles located in the central box.

Using the nearest image criterion it is possible to calculate for every neighbour particle of a given CP its position, i.e. to calculate how the parent particle in the central box should be shifted to become the nearest image. With the new algorithm, instead of shifting neighbouring particles the CP-COSP is shifted. By using the nearest image criterion and the positions of particles in the neighbourlist of a given CP-COSP, it is possible in principle to calculate in which image box this CP-COSP should be located. This however is inefficient. It is much better to identify a CP-COSP by its particle number *and its box identifier* k. Doubling the memory requirements for CP-COSPs is relatively cheap compared to the memory requirements for the neighbourlists. In the rest of this chapter we shall mean by the COSP method the combination of (2.8) and the use of stored box identifiers of CP-COSPs in the CP-COSP list.

There is yet another reason for storing the box identifiers, which has to do with the integrity of charge groups. A charge group is a small group of particles which are involved in a common cut-off criterion, which means that all these particles or none of these particles are involved in a particular interaction. During NBF calculations all particles of a given charge group should undergo the same translation, even when this leads to a violation of the nearest image criterion. That is because a disrupted charge group would create a non-physical electric field over a long distance, leading to erroneous simulation results. The only way to prevent this situation is to calculate during neighbour searching the required translation of the charge group as a whole, and to use during NBF calculations this translation for all particles of the charge group, no matter what their actual position may become during subsequent timesteps.

The identical translation requirement of charge group particles is not introduced by the COSP method. For the same reason as depicted above, in conventional implementations the translation of particles constituting a charge group should also be the same and kept constant between two successive neighbour searching calls.

2.3 The Implementation

2.3.1 The algorithm

In this subsection we will show the outlines of an implementation of the M.D. algorithm with emphasis on the implementation of the new NBF method (2.8) and the new virial method (2.22). We will use a Pascal-like pseudo programming language which is capable of assigning vectors with one statement, returning a vector as a function result, etc. Some parts of the implementation are only outlined while other parts are given in more detail. The best way to understand this program is to first study the neighbourlist structure, that is, the data structures `cp_cosp` and `nl`. To comment on this listing line numbers are used. After the listing the comments are given. To keep the listing clear, we use procedures without parameter list, which implies global access to data. Names used in this pseudo implementation are as close as possible to the symbols in the rest of this text. For the sake of simplicity we do not introduce differing particle types, charge groups, bonded forces, etc.

```
01: program MD;
02: constant
03:     N = 10000; {= maximum nr of particles}
```

```
04:    MAX_NR_CP_COSP = 8*N;
05:    MAX_NR_INTERACTIONS = 160*N;
06: type vec = array [1..3] of real;
07: var
08:    cp_cosp: array [1..MAX_NR_CP_COSP+1] of
09:                                          record i, k, first: integer end;
10:    nl:          array [1..MAX_NR_INTERACTIONS] of integer;
11:    r, v, F:     array [1..N] of vec;
12:    n, nr_cp_cosp, nr_timesteps, lifetime_of_nl, i: integer;
13:    t, g:        array [−13..13] of vec;
14:    w:           array [1..3, 1..3] of real;

15: procedure nbf;
16:    var
17:       a, b, i, j, k:   integer;
18:       f_i_k, fij, r_i: vec;
19:    begin
20:       g := 0;      F := 0;
21:       for a:=1 to nr_cp_cosp do begin
22:          i := cp_cosp[a].i;      k := cp_cosp[a].k;
23:          f_i_k := 0;         r_i := r[i] + t[k];
24:          for b:=cp_cosp[a].first to (cp_cosp[a+1].first)−1 do begin
25:             j:= nl[b];
26:             fij := force(r_i,r[j]);
27:             f_i_k += fij;      F[j] -= fij;
28:          end; {b loop}
29:          g[k] += f_i_k;      F[i] += f_i_k;
30:       end; {a loop}
31:       w := 0;
32:       for i:=1 to n do w += r[i] ⊗ F[i];
33:       for k:=−13 to 13 do w += t[k] ⊗ g[k];
34:    end;{nbf}

35: begin {main}
36:    initialise;
37:    for i:=1 to nr_timesteps do begin
```

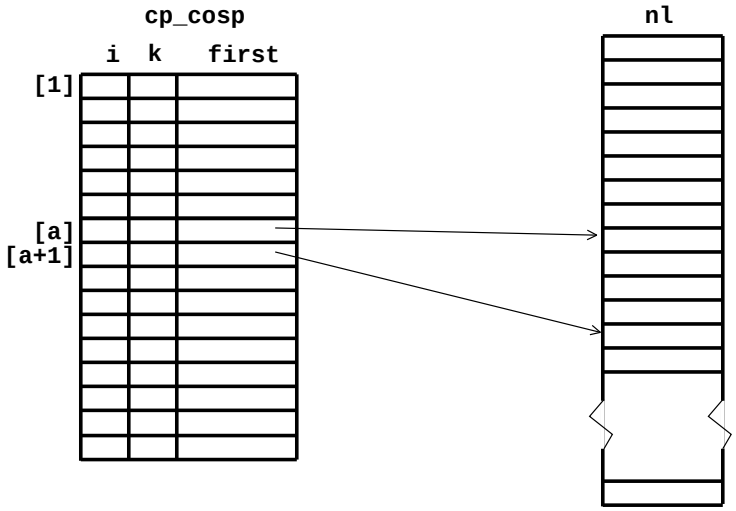

FIGURE 2.3 The neighbourlist structure of our pseudo implementation. In the array **cp-cosp** for every CP-COSP the particle number and box number is stored, as well as an index in the array **nl**. In the array **nl** the neighbours of the CP-COSP **cp_cosp[a]** are located between **cp_cosp[a].first** and **(cp_cosp[a+1].first)-1**.

38: **if** (i **mod** lifetime_of_nl) = 1 **then** search_neighbours;
39: nbf;
40: integrate;
41: **end**; {i loop }
42: **end**; {main}

Comments:
04: We have chosen the maximum number of CP-COSPs to be 8∗**N**, which works fine for small **N**. For large **N**, and a normal R_{co}, there are far less CP-COSPs because then many of the cut-off spheres intersect with less than eight boxes.
05: We assume a maximum average number of particles within R_{co} of 320. This results in an average of at most 160 non-bonded force calculations per particle.
08, 10: For storing neighbour information two arrays are used: **cp_cosp** and **nl**. In **cp_cosp** the particle number i and the box number k of every CP-COSP is stored. The neighbours of the CP-COSP described in **cp_cosp[a]** are located in the array **nl**, at the places with index **cp_cosp[a].first**, ..., **(cp_cosp[a+1].first)-1**. See Figure 2.3.

11: Position, velocity, and total force.
12: Actual number of particles, actual number of CP-COSPs during this lifetime of the neighbourlists, total number of timesteps to be done, number of timesteps to be done with the same neighbour lists, and loop variable.
13, 14: See (1), (20), and (9). **t** is updated every time the box dimensions change, e.g. by pressure scaling.
18: Accumulated force on CP-COSP $i.k$, interaction force introduced for the sake of efficiency, and image position of particle i introduced for the sake of efficiency.
20: Clear the arrays **g** and **F**.
26: The function **force**, not declared in this pseudo program, returns a vector valued result.
32, 33: See (22). We assume the existence of the tensorial product in our pseudo language.
36: Reads in the input, and initialises all relevant variables.
38: The procedure **search_neighbours** is not declared in this pseudo program. Every time it is invoked it fills the variables **cp_cosp**, **nl**, and **nr_cp_cosp**.
40: Using **r**, **v**, and **F**, the variables **r** and **v** are updated.

2.3.2 The implementation

Based on the previous algorithm we have made an implementation of the M.D. algorithm with COSP and, for comparison purposes, an implementation without COSP (non-COSP). Both implementations were done in C.[1] For a fair comparison the implementations only differ at those places where the COSP and non-COSP algorithms differ, that is, in the neighbour searching routine and the NBF routine. The **main** body of both implementations is exactly the same.

In the implementation of both the COSP method and the non-COSP method we store particle numbers *and box identifiers* in the neighbourlists. As we explained before, this means that, in case of COSP, of every CP-COSP its number and box identifier k is stored. In case of non-COSP, the neighbourlist of every CP consists of particle numbers and box identifiers k. This makes our non-COSP implementation somewhat unconventional because in the NBF routine normally the nearest image criterion is used to determine the image box of particles. However, using the nearest image criterion for non-COSP, and the stored box identifier for COSP would lead to

(1) The complete set of sources can be obtained by anonymous ftp from **ftp.cs.rug.nl** in directory **pub/moldyn**.

an unfair comparison between the speeds of the COSP and non-COSP programs in favour of the COSP method. This is because the nearest image calculation is more expensive than the calculation of image positions using stored box identifiers.

To minimise the time required for neighbour searching we use a grid search method. This means that a grid is constructed in the computational box, and that the particles are assigned to the appropriate grid cell before constructing the neighbourlist. Searching for neighbours of a particle is done by only inspecting the particles in its own and neighbouring cells. Experience shows that a grid size $L = \frac{1}{2}R_{co}$ gives an optimal neighbour searching speed. Approximately one in four inspected particles is then in cut-off range. For M.D. systems as used in our tests, using a grid search technique gives an approximate $\frac{NS}{NBF}$ CPU time ratio of $1 \div 3$. (See test results.) Using an all-pairs neighbour searching technique gives a ratio of $2 \div 1$, i.e. the NS time is six times longer without the use of a grid search technique. The COSP method causes no problems with the implementation of the grid search technique.

In order to allow a straightforward interpretation of the results we kept our test programs simple. That means that a single cut-off range was implemented, only one particle type was considered, no charge groups were implemented, no Bonded Forces (BF) were calculated etc.

2.3.3 The machines used

We tested our implementations on a wide variety of machines to show that the type of machine does not influence the speed improvement we get. We also ran our program on the Convex and CM5, but because we did not vectorise or parallelise our code we do not include the results of those runs. We used the following machines and software:

HP1 Hewlett Packard HP 9000-735; 128 Mb memory; 200 Mb swap; gcc 2.3.3.

HP2 Hewlett Packard HP 9000-720; 64 Mb memory; 200 Mb swap; gcc 2.3.3.

IRIS Silicon Graphics SGI 210; 32 Mb memory; 48 Mb swap; gcc 2.4.3.

i860 Intel i860XR; 40 Mhz; 8 Mb memory; hcc.

i486 Intel 80486 running SCO Unix; 66 Mhz; Local bus; 32 Mb memory; 64 Mb swap; gcc 2.4.3.

SUN SPARC station SLC; 16 Mb memory; 40 Mb swap; gcc 2.3.3.

2.3.4 The test M.D. system

All tests were done on an argon-like system with PBC consisting of 8000 particles in a cubic box with sides 21.0 units long (reduced density $\rho^{\star} = 0.863$). The potential used is the dimensionless Lennard-Jones potential

$$V(r) = -4.0/r^6 + 4.0/r^{12} \,. \tag{2.28}$$

For all tests the initial system conditions are identical. Initially particles are distributed over the box by placing them at the grid points of a cubical grid, and giving them small random displacements. Initial velocities are Gaussian distributed with random direction, such that the net system momentum is zero.

R_{co} was chosen such that there were either 100 or 300 neighbours on average within cut-off range. The neighbourlist was updated every 10 timesteps. Having 100 neighbours means that on average approximately 50 NBF evaluations per particle are done. This number of particles within cut-off is typical for particles in a protein, which are only interacting with other protein particles. The other R_{co} is typical of water atoms which are only interacting with other water atoms.

The timestep was the same for all simulations, and all runs had a length of 10 timesteps.

2.3.5 The Test Runs

To keep the test circumstances on all machines as much as possible constant the same operating system (UNIX) and the same files were used for all tests. This means that the same compiler, the gcc compiler, was used for all tests on all machines. The exception was the i860 for which no gcc compiler was available to us. On that machine we used the Intel High C compiler instead.

The timing was done by clocking the two parts of the inner loop using the system clock. Again, the i860 was the exception because in this case the timing was done using the clock of a connected transputer.

We used single processor programs; no use was made of the multiprocessor capabilities some of the machines have.

2.3.6 Results

In Table (2.1) the results of the measurements done for both the non-COSP and COSP program for both cut-offs, with their respective number of neighbours, are given as

the number of steps per second and the percentage of time spent on non-bonded force calculations for each run. It is easy to deduce from the table the execution time of the NBF routine.

	NON-COSP				**COSP**			
neighbours	100		300		100		300	
system	speed	%NBF	speed	%NBF	**speed**	%NBF	**speed**	%NBF
HP1	0.64	(73)	0.21	(76)	**0.89**	(64)	**0.31**	(68)
HP2	0.27	(74)	0.087	(79)	**0.40**	(68)	**0.14**	(72)
IRIS	0.26	(79)	0.083	(82)	**0.38**	(81)	**0.12**	(84)
i860	0.17	(73)	0.050	(72)	**0.25**	(70)	**0.076**	(73)
i486	0.097	(79)	0.030	(82)	**0.16**	(83)	**0.052**	(82)
SUN	0.031	(84)	0.010	(86)	**0.049**	(85)	**0.015**	(87)

TABLE 2.1 Comparison of the speed in steps per sec. of the conventional (non-COSP) and the COSP algorithm, including neighbour searching once per 10 steps, performed on various machines. The tests were done for two cut-off ranges (100 and 300 particles within R_{co}). The COSP implementation is consistently $\approx 1.5\times$ faster than a conventional implementation. In parenthesis the percentage of total step time spent in NBF. The rest of the time was taken up by NS.

From these results it is clear that the COSP implementation is consistently almost 1.5 times faster than the non-COSP implementation. This is the most important conclusion to be drawn from the table.

Some further observations can be made. As expected, on all machines the number of timesteps per second decreases by a factor of three when the number of particles within cut-off range is increased by a factor of three.

The percentage of time spent in NBF calculations is lower for the high performance machines. Because the rest of the time is spent in neighbour searching this means that neighbour searching is relatively more efficient on simpler machines. However, in all tests, irrespective of the machine, the times spent on neighbour searching and on NBF calculations for the COSP implementation are less than the corresponding times for the non-COSP implementation.

The differences in hardware speeds also account for the differences in time spent in

force calculations between COSP and non-COSP. In neighbour searching the COSP way the percentage of integer calculations is much higher than in the non-COSP way.

2.4 Extensions

2.4.1 Generalisation to other box shapes

In the previous sections of this chapter we assumed that the computational box, and consequently image boxes, are triclinic. When we limit ourselves to convex figures there are however five different regular figures, called parallelohedra, whose translated replicas can be fitted together along whole faces to cover the whole space just once [5]. In increasing order of complexity they are: the cube (six squares), the hexagonal prism (six rectangles and two hexagons), the rhombic dodecahedron (twelve rhombi), the elongated dodecahedron (eight parallelograms and four hexagons), and the truncated octahedron (six squares and eight hexagons). Of course the affine transforms of these figures can also be stacked in a space filling way, which for example generalises the cube to the triclinic shape. All five shapes consist of pairwise parallel planes, so for every one L_{min} can be defined analogous to the definition in Section 2.2.1.

Of these five figures the cube requires the largest minimal number of surrounding image systems to be fully enclosed by these image systems. That is because the cube has six faces in common with surrounding systems, twelve vertices, and eight edges, giving a total number of surrounding systems of 26.

The derivations in Section 2.2 were done for a triclinic box but they also hold for the other four box shapes because (2.4) (2.5) and (2.6) do not depend on the box shape. All the information concerning the place of image boxes is stored in **t**. So, when the COSP method is used *literally the same NBF and virial routine may be used for all five box shapes*. For the time being, only the neighbour searching routine is different for every box shape, as is the routine which resets particles in the box previous to neighbour searching. In a next chapter we will however show how these five figures can be mapped on a rectangular box. This will make neighbour searching and resetting particles also generic with respect to the box shape. Combining the COSP method with the generic neighbour searching and resetting algorithm will result then in a completely box shape independent M.D. algorithm.

2.4.2 Applicability

In this section we make a few remarks about other possible applications of the methods we presented in earlier sections of this chapter, and a few remarks about implementing only a part of the methods proposed.

Large cut-off. In the usual M.D. simulation practice $R_{co} < \frac{1}{2}L_{min}$. For a part this is for physical reasons; to mitigate effects introduced by PBC a particle should interact with at most one image of a particle. Another reason is that with existing M.D. software it is difficult to identify multiple images of a given particle. With our method this is no problem because CP-COSPs are identified by a particle number *and a box identifier* k. In that way it is possible to represent multiple occurrences of the same parent particle within R_{co} of a given particle. In our implementation this would only mean a longer **t** and **g** array, and require a modified neighbour searching routine.

Interactions over the central box boundary in a half space. In the usual M.D. simulation practice and also in our implementation particles in the central box interact with particles in the central and surrounding boxes, where surrounding boxes may have both positive and negative image identifiers k. With the notation and transformation rules we introduced in Section 2.2 it is however possible to derive a specification and consequently an implementation in which particles in the central box interact with particles in the central and surrounding boxes, with only non-negative image identifiers k. That is because for every interaction of a particle in the central box with a particle in an image system there exists a translated but otherwise identical interaction over the opposite box boundary (see Figure 2.1). Which interaction is evaluated in the NBF routine is a matter of choice, so the neighbourlists may be constructed in such a way that only interactions which cross the boundary to image systems with a non-negative k are entered into the lists.

At this moment we do not see a useful application for an algorithm of this kind but maybe it is useful on a future exotic parallel computer. We mention an algorithm of this kind to show how useful formal methods as introduced in this chapter are to derive alternative M.D. algorithms.

Mixed COSP and conventional implementations. From practical considerations one may wish to implement only certain parts of the COSP method. For example, one may wish to use (2.22) for calculating the virial in an otherwise conventional implementation. Also, one may wish to use the COSP method without the introduction of explicit box identifiers, or the other way around, to use explicit box

identifiers but not the other parts of the COSP method. Although not impossible in principle, implementations of this kind are inconsistent and clumsy. We recommend that the COSP method should be adopted as a whole or a conventional implementation should be used.

2.5 Conclusion

Specifying and transforming the NBF and virial routine in case of PBC, and the introduction of the explicit image identifier k has roughly speaking three results. First, the formal derivation gives existing and new M.D. algorithms a solid base. Yet, the derivations are so simple that the contact between the derivation and the actual M.D. algorithm is never lost. This gives a better insight in what is going on in the NBF and virial algorithm.

Secondly, the algorithms we derived, are significantly ($1.5\times$) faster than conventional algorithms, without penalty on applicability.

Thirdly, the algorithms are generic with respect to the box shape. The same NBF and virial routine may be used for triclinic, truncated octahedron box shapes etc., without any loss of speed. Only the neighbour searching routine has to know what box shape is being used. The character of the NBF and virial routine is generic because the COSP method strongly reduces the number of images to be calculated, and thus makes it possible to store explicit image identifiers k of the few remaining images instead of recalculating images during the NBF calculations.

In short, using COSP in MD programs will make these programs run faster with roughly a factor 1.5 compared with conventional implementations, and will make the NBF routine general with respect to box shape.

Acknowledgements

We thank H. Keegstra and B. Reitsma for useful discussions, and Biomos b.v. for giving us the opportunity to discuss ideas presented in this chapter at the Burg Arras Biomos '93 meeting.

Literature

[1] H. Bekker, H.J.C. Berendsen, E.J. Dijkstra, S. Achterop, R. v. Drunen, D. v.d. Spoel, A. Sijbers, H. Keegstra, B. Reitsma, and M.K.R. Renardus, in *Conf. Proc. Physics Computing '92*, 257-261, World Scientific Publishing Co. Singapore, New York, London.

[2] H. Bekker, H.J.C. Berendsen, E.J. Dijkstra, S. Achterop, R. v. Drunen, D. v.d. Spoel, A. Sijbers, H. Keegstra, B. Reitsma and M.K.R. Renardus, in *Conf. Proc. Physics Computing '92*, 252-256, World Scientific Publishing Co. Singapore, New York, London.

[3] H. Bekker, E.J. Dijkstra, H.J.C. Berendsen, *Supercomputer* 54, X-2 (1993), 4-10

[4] J.J. Erpenbeck, W.W. Wood, in *Statistical Mechanics B. Modern Theoretical Chemistry*, ed. B.J. Berne (Plenum, New York, 1977), Vol. 6, 1-40.

[5] L. Fejes Tóth, *Regular Figures*, Pergamon Press, London, 1964, 114-119.